**Bài thực hành số 8.**

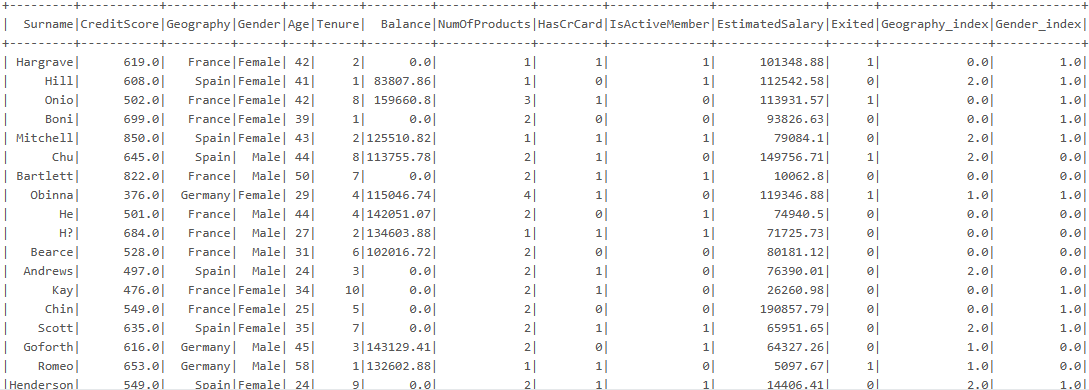
**Tăng hiệu quả cho mô hình học máy**

**Link bài làm trên databricks:** [bài làm](https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/345358637122346/4444005907891880/5017236833160125/latest.html)

**Đọc dữ liệu Churn\_Modelling vào df**

df = spark.read.option('header', 'true').csv('/FileStore/Bigdata\_task07\_dataclean/Churn\_Modelling\_Processed.csv')

df.show()

****

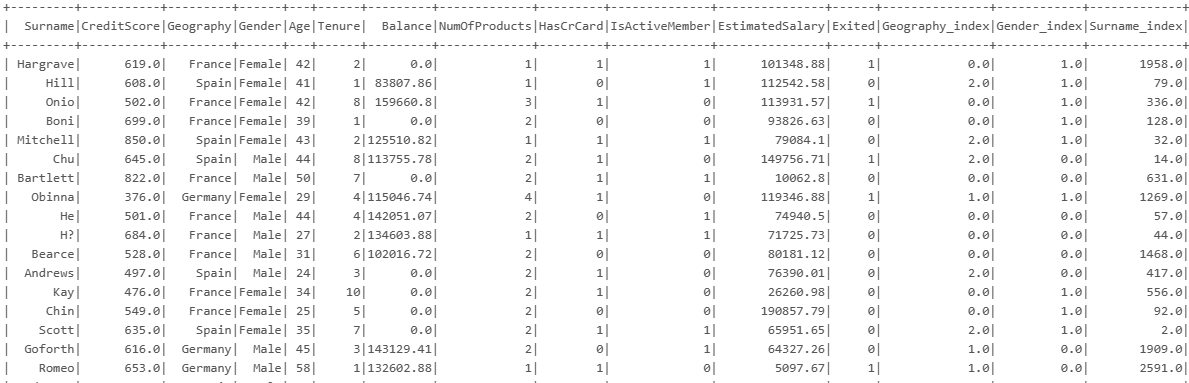
**Đánh index cho cột Surname**

from pyspark.ml.feature import StringIndexer

stringIndexer = StringIndexer(inputCol='Surname', outputCol='Surname\_index')

df = stringIndexer.fit(df).transform(df)

df.show()

****

**Bài 1. Lựa chọn đặc trưng**

**Tạo các cột đặc trưng**

col\_assembler =  ["CreditScore", "Age", "Tenure", "Balance", "NumOfProducts", "HasCrCard", "IsActiveMember", "Geography\_index", "Gender\_index", "Surname\_index"]

assembler = VectorAssembler(inputCols=col\_assembler, outputCol='features')

df\_assembler = assembler.transform(df)

**Trích chọn các đặc trưng bằng ChiSqSelector**

from pyspark.ml.feature import ChiSqSelector

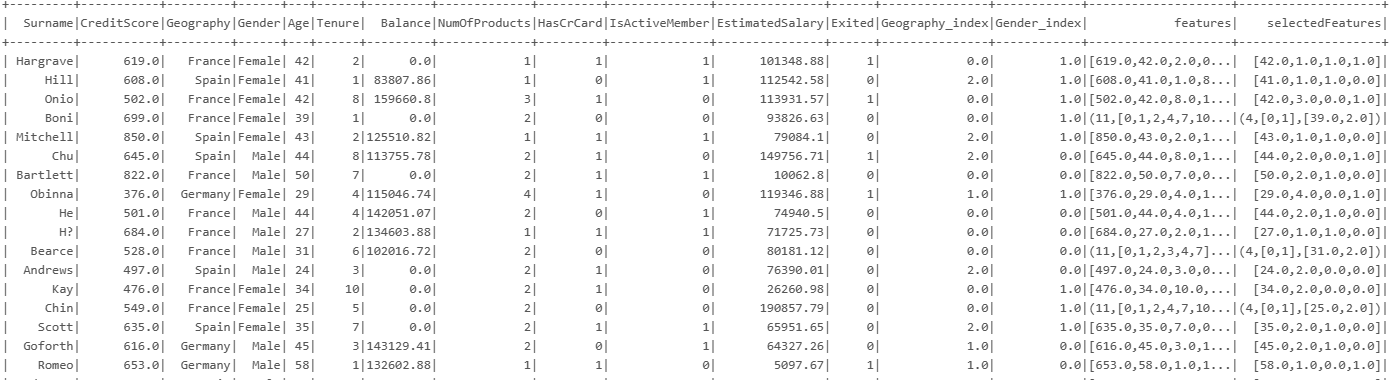
selector = ChiSqSelector(numTopFeatures=4, featuresCol='features', outputCol='selectedFeatures', labelCol='Exited')

css\_model = selector.fit(df\_assembler)

print(css\_model.selectedFeatures) # [1, 4, 6, 7]

data = css\_model.transform(df\_assembler)

data.show()

****

**Xây dựng mô hình bằng thuật toán Hồi Quy Logistic**

from pyspark.ml.classification import LogisticRegression

data\_train, data\_test = data.randomSplit([0.7, 0.3], seed=666)

lr = LogisticRegression(featuresCol='selectedFeatures', labelCol='Exited')

model = lr.fit(data\_train)

**Dự đoán kết quả và kiểm tra độ đo AUC**

from pyspark.ml.evaluation import BinaryClassificationEvaluator

test\_model = model.transform(data\_test)

evaluator\_auc = BinaryClassificationEvaluator(labelCol="Exited", rawPredictionCol="rawPrediction", metricName="areaUnderROC")

auc = evaluator\_auc.evaluate(test\_model)

print("AUC:", auc)

****

**=>** Đánh giá: so kết quả mô hình của bài thực hành trước (không thực hiện trích chọn đặc trưng) với độ đo AUC là **0.9999996441924378** thì ở bài này, sau khi thực hiện trích chọn đặc trưng ở bài này, độ đo AUC của mô hình đạt giá trị thấp hơn là **0.729569063046385**

**Bài 2. Giảm chiều dữ liệu**

**Dùng lớp PCA tính toán 4 thuộc tính quan trọng trong các thuộc tính:**

Surname, CreditScore, Geography, Gender, Age, Tenure, NumOfProducts, HasCrCard,

IsActiveMember, EstimatedSalary.

col\_assembler =  ["CreditScore", "Age", "Tenure", "EstimatedSalary", "NumOfProducts", "HasCrCard", "IsActiveMember", "Geography\_index", "Gender\_index", "Surname\_index"]

assembler = VectorAssembler(inputCols=col\_assembler, outputCol='features')

df\_assembler = assembler.transform(df)

**Dùng lớp PCA tính toán 4 thuộc tính quan trọng**

from pyspark.ml.feature import PCA

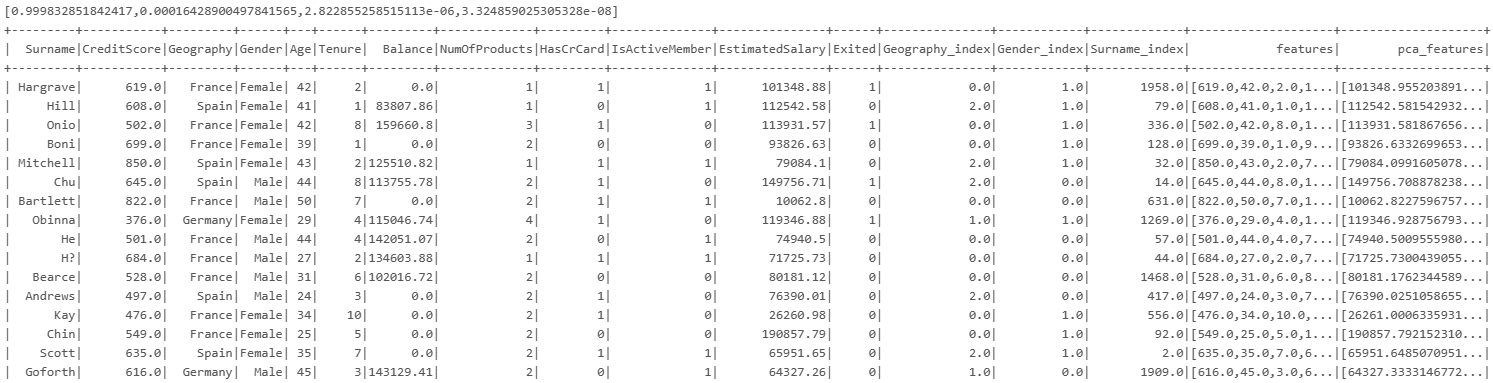
pca = PCA(k=4, inputCol='features', outputCol='pca\_features')

model\_pca = pca.fit(df\_assembler)

print(model\_pca.explainedVariance)

data = model\_pca.transform(df\_assembler)

data.show()

****

**Xây dựng mô hình bằng thuật toán Hồi Quy Logistic**

data\_train, data\_test = data.randomSplit([0.7, 0.3], seed=666)

lr = LogisticRegression(featuresCol='pca\_features', labelCol='Exited')

model = lr.fit(data\_train)

**Dự đoán kết quả và kiểm tra độ đo AUC**

test\_model = model.transform(data\_test)

evaluator\_auc = BinaryClassificationEvaluator(labelCol="Exited", rawPredictionCol="rawPrediction", metricName="areaUnderROC")

auc = evaluator\_auc.evaluate(test\_model)

print("AUC:", auc)

****

**=>** So với kết quả với bài trên là **0.729569063046385** thì kết quả có chính xác một ít. Nhưng so với bài trường thì độ chính xác thấp hơn nhiều.

**Bài 3. Điều chỉnh tham số**

Xây dựng lưới tham số cho 2 tham số maxIter trong danh sách [5, 10, 20] và paramReg

trong danh sách [0.02, 0.05, 0.1] cho mô hình học máy hồi quy logistic dự đoán khách hàng rời ngân hàng trên dữ liệu đã cho với các thuộc tính đã chọn ở Bài 1.

param\_grid = ParamGridBuilder()

param\_grid.addGrid(lr.maxIter, [5, 10, 20])

param\_grid.addGrid(lr.regParam, [0.02, 0.05, 0.1])

grid = param\_grid.build()

**Sử dụng lớp TrainValidationSplit để lựa chọn bộ tham số tốt nhất cho mô hình học máy.**

from pyspark.ml.tuning import TrainValidationSplit

tvs = TrainValidationSplit(estimator = lr, estimatorParamMaps = grid, evaluator = evaluator, trainRatio = 0.8, seed = 123)

model\_tvs = tvs.fit(data\_train)

**Đánh giá độ đo AUC của mô hình tối ưu so với kết quả của Bài 1.**

best\_model = model\_tvs.bestModel

test\_model\_tvs = best\_model.transform(data\_test)

auc = evaluator.evaluate(test\_model\_tvs)

print("AUC on test data = %g" % auc)

****

**=>** So với độ đo ở bài 1 là **0.729569063046385** thì độ đo AUC có nhỉnh hơn 1 tí.

**Cho biết bộ tham số tối ưu là tham số nào.**

regParam = best\_model.getOrDefault("regParam")

maxIter = best\_model.getOrDefault("maxIter")

print(f"regParam: {regParam}")

print(f"maxIter: {maxIter}")

****

**Thay lớp TrainValidationSplit bằng lớp CrossValidator và kiểm tra độ đo và bộ tham**

**số tối ưu có thay đổi không?**

from pyspark.ml.tuning import CrossValidator

crossval = CrossValidator(estimator = lr, estimatorParamMaps = grid, evaluator = evaluator, numFolds=5)

**Kiểm tra độ đo có thay đổi không**

model\_crs = crossval.fit(data\_train)

best\_model = model\_crs.bestModel

test\_model\_crs = best\_model.transform(data\_test)

auc = evaluator.evaluate(test\_model\_crs)

print("AUC on test data = %g" % auc)

****

**=>** Độ đo không thay đổi.

**Kiểm tra bộ tham số tối ưu có thay đổi không**

regParam = best\_model.getOrDefault("regParam")

maxIter = best\_model.getOrDefault("maxIter")

print(f"regParam: {regParam}")

print(f"maxIter: {maxIter}")

****

**=>** Bộ tham số tối ưu không thay đổi.